Dessiner une droite

Le plan est muni d'un repère orthonormé $(O;\vec{i},\vec{j})$. On considère deux points **distincts** A et B, de coordonnées respectives $(x_A;y_A)$ et $(x_B;y_B)$ et on note $dx=x_B-x_A$ et $dy=y_B-y_A$ les différences entre les coordonnées de A et B. Si dx est non nul, on note $m=\frac{dy}{dx}$ la pente de la droite (AB).

On suppose pour toute la suite que A et B sont à coordonnées entières positives et de plus que $x_A\leq x_B$ et $y_A\leq y_B$. (Noter que dx et dy sont donc des entiers.)

Q1. Ecrire une fonction **pgcd(a,b)** qui renvoie le pgcd de deux entiers positifs non nuls a et b. On peut utiliser l'algorithme d'Euclide mais ce n'est pas une obligation.

Q2. Ecrire une fonction **simplifie(a,b)** qui prend en arguments deux entiers strictement positifs a et b, et renvoie les nombres a/d et b/d où d est le pgcd de a et b. La fonction **simplifie** utilisera la fonction **pgcd**.

Q3. On considère la fonction **point_exact** définie ci-dessous.

```
def point_exact(xA,yA,xB,yB):
    dx=xB-xA
    dy=yB-yA
    if dx!=0 and dy!=0:
        dx,dy=simplifie(dx,dy)
        return xA+dx,yA+dy
    elif dx==0:
        return xA,yA+1
    else:
        return xA+1,yA
```

Expliquer précisément ce que représentent les deux nombres renvoyés par cette fonction ?

Qu'affiche l'instruction `print(point_exact(2,1,6,3))` ?
Qu'affiche l'instruction `print(point_exact(2,1,2,7))` ?
Qu'affiche l'instruction `print(point_exact(2,1,6,1))` ?

Q4. Soit f la fonction définie par $f(x,y)=dy(x-x_A)-dx(y-y_A)$. L'équation $f(x,y)=0$ est une équation cartésienne de la droite (AB).

Ecrire une fonction **equation_droite(xA,yA,xB,yB,x,y)** qui renvoie la valeur de $f(x,y)$.

Q5. Quelle est la valeur renvoyée par `equation_droite(2,1,6,3,4,2)` ?
Quelle est la valeur renvoyée par `equation_droite(2,1,6,3,3,2)` ?
Quelle est la valeur renvoyée par `equation_droite(2,1,6,3,5,2)` ?

Le signe et la valeur de $f(x,y)$ permettent de déterminer la position d'un point par rapport à la droite.

L'objectif est de dessiner un segment de droite sur un écran rectangulaire composé de pixels. Chaque pixel est un petit carré de côté 1 unité, considéré comme un point à coordonnées entières positives dans un repère orthonormé dont l'origine, (le pixel de coordonnées $(0;0)$), est en bas à gauche de l'écran. L'ensemble des pixels composent un quadrillage de l'écran.

Q6. Si l'abscisse et l'ordonnée d'un pixel sont codées chacune en binaire sur 12 bits, quel est le nombre maximal de pixels que l'on peut coder ?

Q7. Pour dessiner un segment, on colorie chaque pixel de ce segment en niveau de gris. Ce niveau de gris est codé par un nombre variant de 0 à 255. On stocke en mémoire les triplets (x, y, c) (les coordonnées codées chacune sur 12 bits et la couleur de chaque pixel codée sur 8 bits). Quelle est la taille en octets de la place occupée en mémoire par le stockage des triplets représentant un segment de 500 pixels ?

Q8. On définit une fonction **arrondi** de la manière suivante (**int** renvoie la "partie entière") :

```
def arrondi(x):
    return int(x+0.5)
```

Que renvoie `arrondi(2.6)` ? Que renvoie `arrondi(5.3)` ?

Q9. On considère une droite (AB) de pente comprise entre 0 et 1 et on souhaite dessiner le segment $[AB]$ avec une épaisseur égale à 1, c'est-à-dire que sur chaque verticale, on veut obtenir un seul point ou pixel. La fonction **dessin_segment** permet de construire la liste des coordonnées des pixels permettant de dessiner un segment d'un point $A(x_0, y_0)$ à un point $B(x_0 + dx, y_0 + dy)$.

```
def dessin_segment(x0,y0,dx,dy):
    pixels=[]
    m=dy/dx
    for i in range(dx+1):
        x=x0+i
        y=arrondi(y0+m*i)
        pixels.append((x,y))
    return pixels
```

Que renvoie `dessin_segment(2,1,4,2)` ?

Q10. Exprimer en fonction de dx le nombre total de multiplications, d'additions et d'arrondis effectués lors de l'appel de la fonction **dessin_segment**.

Q11. Quel est le type de la variable m dans la fonction **dessin_segment** ?

Q12. Comment modifier la fonction **dessin_segment** afin de supprimer les multiplications par m ?

Inversement, si on donne maintenant une liste de couples $[(x_0, y_0), (x_1, y_1), \ldots,]$ renvoyée par la fonction **dessin_segment**, donc une liste de couples de coordonnées représentant un segment, on souhaite déterminer la pente de la droite portant ce segment.

Q13. Ecrire une fonction **pente(segment)** qui prend en argument une liste de couples de coordonnées représentant un segment comme ci-dessus et renvoie la pente de la droite portant ce segment.

Pour traiter le cas d'une pente positive supérieure à 1, en utilisant la fonction **dessin_segment** précédente, on considère la symétrie par rapport à la droite $x = y$.

$A(x_A; y_A)$ et $B(x_B; y_B)$ ont pour symétriques respectifs $A'(y_A; x_A)$ et $B'(y_B; x_B)$. On appelle la fonction **dessin_segment** avec A' et B' et il reste alors à prendre les symétriques des points renvoyés par la fonction **dessin_segment** dans la liste **pixels**.

Q14. Ecrire une fonction **sym(segment)** qui prend en argument une liste de couples (x, y) et renvoie la liste des couples (y, x). Par exemple si la liste est [(2, 1), (3, 2), (4, 2), (5, 3), (6, 3)], la fonction renvoie la liste [(1, 2), (2, 3), (2, 4), (3, 5), (3, 6)].

On utiliserait le même principe pour une droite de pente négative mais cette fois avec une symétrie par rapport à la droite d'équation $y = y_A$.